

Welcome to [E-XFL.COM](#)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Obsolete
Core Processor	HCS12
Core Size	16-Bit
Speed	25MHz
Connectivity	CANbus, EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	35
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12c128cpber">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12c128cpber</a>

## Chapter 2

# Port Integration Module (PIM9C32) Block Description

### 2.1 Introduction

The Port Integration Module establishes the interface between the peripheral modules and the I/O pins for all ports.

This chapter covers:

- Port A, B, and E related to the core logic and the multiplexed bus interface
- Port T connected to the TIM module (PWM module can be routed to port T as well)
- Port S connected to the SCI module
- Port M associated to the MSCAN and SPI module
- Port P connected to the PWM module, external interrupt sources available
- Port J pins can be used as external interrupt sources and standard I/O's

The following I/O pin configurations can be selected:

- Available on all I/O pins:
  - Input/output selection
  - Drive strength reduction
  - Enable and select of pull resistors
- Available on all Port P and Port J pins:
  - Interrupt enable and status flags

The implementation of the Port Integration Module is device dependent.

#### 2.1.1 Features

A standard port has the following minimum features:

- Input/output selection
- 5-V output drive with two selectable drive strength
- 5-V digital and analog input
- Input with selectable pull-up or pull-down device

Optional features:

- Open drain for wired-OR connections
- Interrupt inputs with glitch filtering

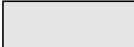
when mapping PWM channels to Port T in an 80QFP option, the associated PWM channels are then mapped to both Port P and Port T.

### 4.3.2.8 Port E Assignment Register (PEAR)

Module Base + 0x000A

Starting address location affected by INITRG register setting.

	7	6	5	4	3	2	1	0
R	NOACCE	0	PIPOE	NECLK	LSTRE	RDWE	0	0
W								
Reset								
Special Single Chip	0	0	0	0	0	0	0	0
Special Test	0	0	1	0	1	1	0	0
Peripheral	0	0	0	0	0	0	0	0
Emulation Expanded Narrow	1	0	1	0	1	1	0	0
Emulation Expanded Wide	1	0	1	0	1	1	0	0
Normal Single Chip	0	0	0	1	0	0	0	0
Normal Expanded Narrow	0	0	0	0	0	0	0	0
Normal Expanded Wide	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 4-12. Port E Assignment Register (PEAR)**

Read: Anytime (provided this register is in the map).

Write: Each bit has specific write conditions. Please refer to the descriptions of each bit on the following pages.

Port E serves as general-purpose I/O or as system and bus control signals. The PEAR register is used to choose between the general-purpose I/O function and the alternate control functions. When an alternate control function is selected, the associated DDRE bits are overridden.

The reset condition of this register depends on the mode of operation because bus control signals are needed immediately after reset in some modes. In normal single-chip mode, no external bus control signals are needed so all of port E is configured for general-purpose I/O. In normal expanded modes, only the E clock is configured for its alternate bus control function and the other bits of port E are configured for general-purpose I/O. As the reset vector is located in external memory, the E clock is required for this access.  $R/\overline{W}$  is only needed by the system when there are external writable resources. If the normal expanded system needs any other bus control signals, PEAR would need to be written before any access that needed the additional signals. In special test and emulation modes, IPIPE1, IPIPE0, E,  $\overline{LSTRB}$ , and  $R/\overline{W}$  are configured out of reset as bus control signals.

This register is not in the on-chip memory map in expanded and special peripheral modes. Therefore, these accesses will be echoed externally.

The interrupt sub-block decodes the priority of all system exception requests and provides the applicable vector for processing the exception. The INT supports I-bit maskable and X-bit maskable interrupts, a non-maskable unimplemented opcode trap, a non-maskable software interrupt (SWI) or background debug mode request, and three system reset vector requests. All interrupt related exception requests are managed by the interrupt sub-block (INT).

### 5.1.1 Features

The INT includes these features:

- Provides two to 122 I-bit maskable interrupt vectors (0xFF00–0xFFF2)
- Provides one X-bit maskable interrupt vector (0xFFF4)
- Provides a non-maskable software interrupt (SWI) or background debug mode request vector (0xFFF6)
- Provides a non-maskable unimplemented opcode trap (TRAP) vector (0xFFF8)
- Provides three system reset vectors (0xFFFA–0xFFFE) (reset, CMR, and COP)
- Determines the appropriate vector and drives it onto the address bus at the appropriate time
- Signals the CPU that interrupts are pending
- Provides control registers which allow testing of interrupts
- Provides additional input signals which prevents requests for servicing I and X interrupts
- Wakes the system from stop or wait mode when an appropriate interrupt occurs or whenever  $\overline{\text{XIRQ}}$  is active, even if  $\overline{\text{XIRQ}}$  is masked
- Provides asynchronous path for all I and X interrupts, (0xFF00–0xFFF4)
- (Optional) selects and stores the highest priority I interrupt based on the value written into the HPRIO register

### 5.1.2 Modes of Operation

The functionality of the INT sub-block in various modes of operation is discussed in the subsections that follow.

- **Normal operation**  
The INT operates the same in all normal modes of operation.
- **Special operation**  
Interrupts may be tested in special modes through the use of the interrupt test registers.
- **Emulation modes**  
The INT operates the same in emulation modes as in normal modes.
- **Low power modes**  
See [Section 5.4.1, “Low-Power Modes,”](#) for details

### 6.3.2.2 BDM CCR Holding Register (BDMCCR)

0xFF06

	7	6	5	4	3	2	1	0
R	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-4. BDM CCR Holding Register (BDMCCR)

Read: All modes

Write: All modes

#### NOTE

When BDM is made active, the CPU stores the value of the CCR register in the BDMCCR register. However, out of special single-chip reset, the BDMCCR is set to 0xD8 and not 0xD0 which is the reset value of the CCR register.

When entering background debug mode, the BDM CCR holding register is used to save the contents of the condition code register of the user's program. It is also used for temporary storage in the standard BDM firmware mode. The BDM CCR holding register can be written to modify the CCR value.

### 6.3.2.3 BDM Internal Register Position Register (BDMINR)

0xFF07

	7	6	5	4	3	2	1	0
R	0	REG14	REG13	REG12	REG11	0	0	0
W								
Reset	0	0	0	0	0	0	0	0


 = Unimplemented or Reserved

Figure 6-5. BDM Internal Register Position (BDMINR)

Read: All modes

Write: Never

Table 6-4. BDMINR Field Descriptions

Field	Description
6:3 REG[14:11]	<b>Internal Register Map Position</b> — These four bits show the state of the upper five bits of the base address for the system's relocatable register block. BDMINR is a shadow of the INITRG register which maps the register block to any 2K byte space within the first 32K bytes of the 64K byte address space.

### 7.3.2.9 Debug Comparator A Extended Register (DBGCAx)

Module Base + 0x002A

Starting address location affected by INITRG register setting.

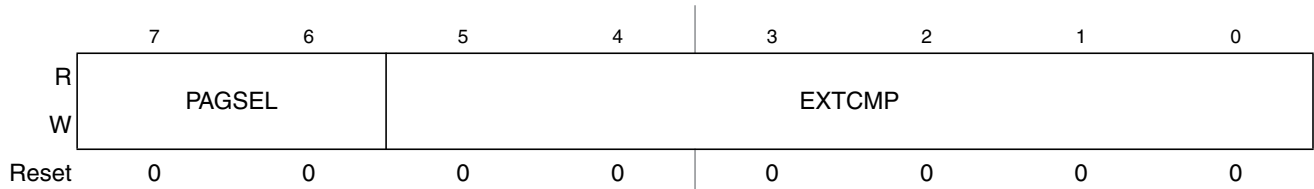


Figure 7-15. Debug Comparator A Extended Register (DBGCAx)

Table 7-19. DBGCAx Field Descriptions

Field	Description
7:6 PAGSEL	<b>Page Selector Field</b> — If DBGGEN is set in DBGCR1, then PAGSEL selects the type of paging as shown in <a href="#">Table 7-20</a> . DPAGE and EPAGE are not yet implemented so the value in bit 7 will be ignored (i.e., PAGSEL values of 10 and 11 will be interpreted as values of 00 and 01, respectively). In BKP mode, PAGSEL has no meaning and EXTCMP[5:0] are compared to address bits [19:14] if the address is in the FLASH/ROM memory space.
5:0 EXTCMP	<b>Comparator A Extended Compare Bits</b> — The EXTCMP bits are used as comparison address bits as shown in <a href="#">Table 7-20</a> along with the appropriate PPAGE, DPAGE, or EPAGE signal from the core.

Table 7-20. Comparator A or B Compares

Mode		EXTCMP Compare	High-Byte Compare
BKP <sup>(1)</sup>	Not FLASH/ROM access	No compare	DBGCRH[7:0] = AB[15:8]
	FLASH/ROM access	EXTCMP[5:0] = XAB[19:14]	DBGCRH[5:0] = AB[13:8]
DBG <sup>(2)</sup>	PAGSEL = 00	No compare	DBGCRH[7:0] = AB[15:8]
	PAGSEL = 01	EXTCMP[5:0] = XAB[21:16]	DBGCRH[7:0] = XAB[15:14], AB[13:8]

1. See [Figure 7-16](#).

2. See [Figure 7-10](#) (note that while this figure provides extended comparisons for comparator C, the figure also pertains to comparators A and B in DBG mode only).

Table 8-10. Available Result Data Formats

SRES8	DJM	DSGN	Result Data Formats Description and Bus Bit Mapping
1	0	0	8-bit / left justified / unsigned — bits 8–15
1	0	1	8-bit / left justified / signed — bits 8–15
1	1	X	8-bit / right justified / unsigned — bits 0–7
0	0	0	10-bit / left justified / unsigned — bits 6–15
0	0	1	10-bit / left justified / signed — bits 6–15
0	1	X	10-bit / right justified / unsigned — bits 0–9

Table 8-11. Left Justified, Signed, and Unsigned ATD Output Codes.

Input Signal $V_{RL} = 0$ Volts $V_{RH} = 5.12$ Volts	Signed 8-Bit Codes	Unsigned 8-Bit Codes	Signed 10-Bit Codes	Unsigned 10-Bit Codes
5.120 Volts	7F	FF	7FC0	FFC0
5.100	7F	FF	7F00	FF00
5.080	7E	FE	7E00	FE00
2.580	01	81	0100	8100
2.560	00	80	0000	8000
2.540	FF	7F	FF00	7F00
0.020	81	01	8100	0100
0.000	80	00	8000	0000

Table 8-12. Analog Input Channel Select Coding

CC	CB	CA	Analog Input Channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7



Table 9-2. CRGFLG Field Descriptions (continued)

Field	Description
1 SCMIF	<b>Self-Clock Mode Interrupt Flag</b> — SCMIF is set to 1 when SCM status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (SCMIE=1), SCMIF causes an interrupt request. 0 No change in SCM bit. 1 SCM bit has changed.
0 SCM	<b>Self-Clock Mode Status Bit</b> — SCM reflects the current clocking mode. Writes have no effect. 0 MCU is operating normally with OSCCLK available. 1 MCU is operating in self-clock mode with OSCCLK in an unknown state. All clocks are derived from PLLCLK running at its minimum frequency $f_{SCM}$ .

### 9.3.2.5 CRG Interrupt Enable Register (CRGINT)

This register enables CRG interrupt requests.

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	RTIE	0	0	LOCKIE	0	0	SCMIE	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-8. CRG Interrupt Enable Register (CRGINT)

Read: anytime

Write: anytime

Table 9-3. CRGINT Field Descriptions

Field	Description
7 RTIE	<b>Real-Time Interrupt Enable Bit</b> 0 Interrupt requests from RTI are disabled. 1 Interrupt will be requested whenever RTIF is set.
4 LOCKIE	<b>Lock Interrupt Enable Bit</b> 0 LOCK interrupt requests are disabled. 1 Interrupt will be requested whenever LOCKIF is set.
1 SCMIE	<b>Self-Clock Mode Interrupt Enable Bit</b> 0 SCM interrupt requests are disabled. 1 Interrupt will be requested whenever SCMIF is set.

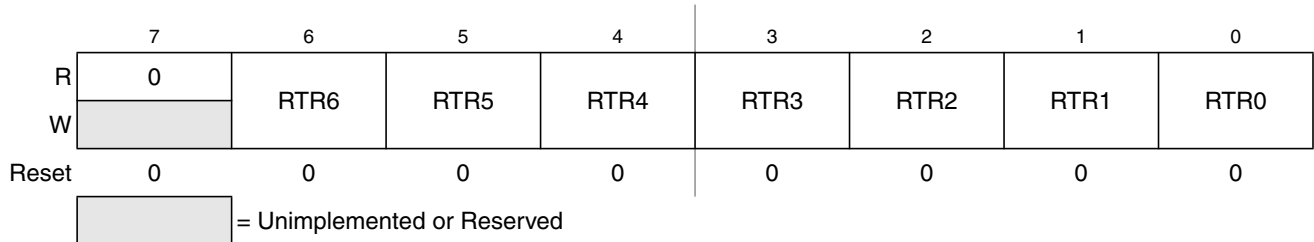
**Table 9-5. PLLCTL Field Descriptions (continued)**

Field	Description
5 AUTO	<b>Automatic Bandwidth Control Bit</b> — AUTO selects either the high bandwidth (acquisition) mode or the low bandwidth (tracking) mode depending on how close to the desired frequency the VCO is running. Write anytime except when PLLWAI=1, because PLLWAI sets the AUTO bit to 1. 0 Automatic mode control is disabled and the PLL is under software control, using ACQ bit. 1 Automatic mode control is enabled and ACQ bit has no effect.
4 ACQ	<b>Acquisition Bit</b> — Write anytime. If AUTO=1 this bit has no effect. 0 Low bandwidth filter is selected. 1 High bandwidth filter is selected.
2 PRE	<b>RTI Enable during Pseudo-Stop Bit</b> — PRE enables the RTI during pseudo-stop mode. Write anytime. 0 RTI stops running during pseudo-stop mode. 1 RTI continues running during pseudo-stop mode. <b>Note:</b> If the PRE bit is cleared the RTI dividers will go static while pseudo-stop mode is active. The RTI dividers will <u>not</u> initialize like in wait mode with RTIWAI bit set.
1 PCE	<b>COP Enable during Pseudo-Stop Bit</b> — PCE enables the COP during pseudo-stop mode. Write anytime. 0 COP stops running during pseudo-stop mode 1 COP continues running during pseudo-stop mode <b>Note:</b> If the PCE bit is cleared the COP dividers will go static while pseudo-stop mode is active. The COP dividers will <u>not</u> initialize like in wait mode with COPWAI bit set.
0 SCME	<b>Self-Clock Mode Enable Bit</b> — Normal modes: Write once —Special modes: Write anytime — SCME can not be cleared while operating in self-clock mode (SCM=1). 0 Detection of crystal clock failure causes clock monitor reset (see <a href="#">Section 9.5.1, “Clock Monitor Reset”</a> ). 1 Detection of crystal clock failure forces the MCU in self-clock mode (see <a href="#">Section 9.4.7.2, “Self-Clock Mode”</a> ).

### 9.3.2.8 CRG RTI Control Register (RTICTL)

This register selects the timeout period for the real-time interrupt.

Module Base + 0x0007



**Figure 9-11. CRG RTI Control Register (RTICTL)**

Read: anytime

Write: anytime

#### NOTE

A write to this register initializes the RTI counter.

Table 10-1. CANCTL0 Register Field Descriptions (continued)

Field	Description
1 SLPRQ <sup>(5)</sup>	<p><b>Sleep Mode Request</b> — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see <a href="#">Section 10.4.5.4, “MSCAN Sleep Mode”</a>). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPK = 1 (see <a href="#">Section 10.3.2.2, “MSCAN Control Register 1 (CANCTL1)”</a>). SLPRQ cannot be set while the WUPE flag is set (see <a href="#">Section 10.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”</a>). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself.</p> <p>0 Running — The MSCAN functions normally</p> <p>1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle</p>
0 INITRQ <sup>(6),(7)</sup>	<p><b>Initialization Mode Request</b> — When this bit is set by the CPU, the MSCAN skips to initialization mode (see <a href="#">Section 10.4.5.5, “MSCAN Initialization Mode”</a>). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (<a href="#">Section 10.3.2.2, “MSCAN Control Register 1 (CANCTL1)”</a>).</p> <p>The following registers enter their hard reset state and restore their default values: CANCTL0<sup>(8)</sup>, CANRFLG<sup>(9)</sup>, CANRIER<sup>(10)</sup>, CANTFLG, CANTIER, CANTARQ, CANTAOK, and CANTBSEL.</p> <p>The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode.</p> <p>When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits.</p> <p>Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0.</p> <p>0 Normal operation</p> <p>1 MSCAN in initialization mode</p>

1. The MSCAN must be in normal mode for this bit to become set.
2. See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.
3. In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see [Section 10.4.5.2, “Operation in Wait Mode”](#) and [Section 10.4.5.3, “Operation in Stop Mode”](#)).
4. The CPU has to make sure that the WUPE register and the WUPE wake-up interrupt enable register (see [Section 10.3.2.6, “MSCAN Receiver Interrupt Enable Register \(CANRIER\)”](#)) is enabled, if the recovery mechanism from stop or wait is required.
5. The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPK = 1).
6. The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).
7. In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPK = 1) before requesting initialization mode.
8. Not including WUPE, INITRQ, and SLPRQ.
9. TSTAT1 and TSTAT0 are not affected by initialization mode.
10. RSTAT1 and RSTAT0 are not affected by initialization mode.

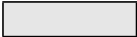
### 10.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

## 12.3.2 Register Descriptions

The following paragraphs describe in detail all the registers and register bits in the PWM8B6CV1 module.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 PWME	R	0	0	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
	W								
0x0001 PWMPOL	R	0	0	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
	W								
0x0002 PWMCLK	R	0	0	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
	W								
0x0003 PWMPRCLK	R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
	W								
0x0004 PWMCAE	R	0	0	CAE5	CAE4	CAE2	CAE2	CAE1	CAE0
	W								
0x0005 PWMCTL	R	0	CON45	CON23	CON01	PSWAI	PFRZ	0	0
	W								
0x0006 PWMTST	R	0	0	0	0	0	0	0	0
	W								
0x0007 PWMPRSC	R	0	0	0	0	0	0	0	0
	W								
0x0008 PWMSCLA	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0009 PWMSCLB	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x000A PWMSCNTA	R	0	0	0	0	0	0	0	0
	W								
0x000B PWMSCNTB	R	0	0	0	0	0	0	0	0
	W								
0x000C PWMCNT0	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x000D PWMCNT1	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0
0x000E PWMCNT2	R	Bit 7	6	5	4	3	2	1	Bit 0
	W	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

**Figure 12-2. PWM Register Summary**

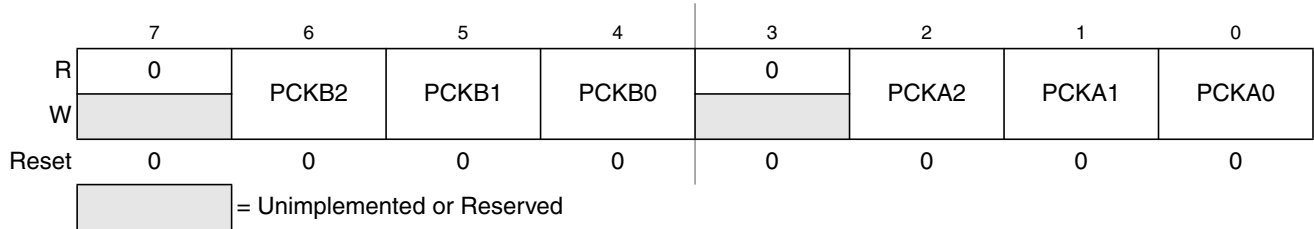
**Table 12-4. PWMCLK Field Descriptions**

Field	Description
5 PCLK5	<b>Pulse Width Channel 5 Clock Select</b> 0 Clock A is the clock source for PWM channel 5. 1 Clock SA is the clock source for PWM channel 5.
4 PCLK4	<b>Pulse Width Channel 4 Clock Select</b> 0 Clock A is the clock source for PWM channel 4. 1 Clock SA is the clock source for PWM channel 4.
3 PCLK3	<b>Pulse Width Channel 3 Clock Select</b> 0 Clock B is the clock source for PWM channel 3. 1 Clock SB is the clock source for PWM channel 3.
2 PCLK2	<b>Pulse Width Channel 2 Clock Select</b> 0 Clock B is the clock source for PWM channel 2. 1 Clock SB is the clock source for PWM channel 2.
1 PCLK1	<b>Pulse Width Channel 1 Clock Select</b> 0 Clock A is the clock source for PWM channel 1. 1 Clock SA is the clock source for PWM channel 1.
0 PCLK0	<b>Pulse Width Channel 0 Clock Select</b> 0 Clock A is the clock source for PWM channel 0. 1 Clock SA is the clock source for PWM channel 0.

### 12.3.2.4 PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently.

Module Base + 0x0003



**Figure 12-6. PWM Prescaler Clock Select Register (PWMPRCLK)**

Read: anytime

Write: anytime

#### NOTE

PCKB2–PCKB0 and PCKA2–PCKA0 register bits can be written anytime. If the clock prescale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Write: Used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a zero will not affect current status of the bit.

**Table 15-17. TRLG1 Field Descriptions**

Field	Description
7:0 C[7:0]F	<b>Input Capture/Output Compare Channel “x” Flag</b> — These flags are set when an input capture or output compare event occurs. Clearing requires writing a one to the corresponding flag bit when TEN is set to one. When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel (0x0010–0x001F) will cause the corresponding channel flag CxF to be cleared.

### 15.3.2.13 Main Timer Interrupt Flag 2 (TFLG2)

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	TOF	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0
	Unimplemented or Reserved							

**Figure 15-21. Main Timer Interrupt Flag 2 (TFLG2)**

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write the bit to one while TEN of TSCR1 is set to one.

Read: Anytime

Write: Used in clearing mechanism (set bits cause corresponding bits to be cleared).

Any access to TCNT will clear TFLG2 register if the TFFCA bit in TSCR register is set.

**Table 15-18. TRLG2 Field Descriptions**

Field	Description
7 TOF	<b>Timer Overflow Flag</b> — Set when 16-bit free-running timer overflows from 0xFFFF to 0x0000. Clearing this bit requires writing a one to bit 7 of TFLG2 register while TEN bit of TSCR1 is set to one. (See also TCRC control bit explanation.)

## Chapter 17

# 16 Kbyte Flash Module (S12FTS16KV1)

### 17.1 Introduction

The **FTS16K** module implements a 16 Kbyte Flash (nonvolatile) memory. The Flash memory contains one array of 16 Kbytes organized as 256 rows of 64 bytes with an erase sector size of eight rows (512 bytes). The Flash array may be read as either bytes, aligned words, or misaligned words. Read access time is one bus cycle for byte and aligned word, and two bus cycles for misaligned words.

The Flash array is ideal for program and data storage for single-supply applications allowing for field reprogramming without requiring external voltage sources for program or erase. Program and erase functions are controlled by a command driven interface. The Flash module supports both mass erase and sector erase. An erased bit reads 1 and a programmed bit reads 0. The high voltage required to program and erase is generated internally. It is not possible to read from a Flash array while it is being erased or programmed.

#### CAUTION

A Flash word must be in the erased state before being programmed.  
Cumulative programming of bits within a Flash word is not allowed.

#### 17.1.1 Glossary

**Command Write Sequence** — A three-step MCU instruction sequence to program, erase, or erase verify the Flash array memory.

#### 17.1.2 Features

- 16 Kbytes of Flash memory comprised of one 16 Kbyte array divided into 32 sectors of 512 bytes
- Automated program and erase algorithm
- Interrupts on Flash command completion and command buffer empty
- Fast sector erase and word program operation
- 2-stage command pipeline for faster multi-word program times
- Flexible protection scheme to prevent accidental program or erase
- Single power supply for Flash program and erase operations
- Security feature to prevent unauthorized access to the Flash array memory

## 17.3.2 Register Descriptions

The Flash module contains a set of 16 control and status registers located between module base + 0x0000 and 0x000F. A summary of the Flash module registers is given in Figure 17-3. Detailed descriptions of each register bit are provided.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 FCLKDIV	R	FDIVLD	PRDIV8	FDIV5	FDIV4	FDIV3	FDIV2	FDIV1	FDIV0
	W								
0x0001 FSEC	R	KEYEN1	KEYEN0	NV5	NV4	NV3	NV2	SEC1	SEC0
	W								
0x0002 RESERVED1 <sup>(1)</sup>	R	0	0	0	0	0	0	0	0
	W								
0x0003 FCNFG	R	CBEIE	CCIE	KEYACC	0	0	0	0	0
	W								
0x0004 FPROT	R	FPOPEN	NV6	FPHDIS	FPHS1	FPHS0	NV2	NV1	NV0
	W								
0x0005 FSTAT	R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	FAIL	DONE
	W								
0x0006 FCMD	R	0	CMDB6	CMDB5	0	0	CMDB2	0	CMDB0
	W								
0x0007 RESERVED2 <sup>1</sup>	R	0	0	0	0	0	0	0	0
	W								
0x0008 FADDRHI <sup>1</sup>	R	0	0	0	FABHI				
	W								
0x0009 FADDRLO <sup>1</sup>	R	FABLO							
	W								
0x000A FDATAHI <sup>1</sup>	R	FDHI							
	W								
0x000B FDATALO <sup>1</sup>	R	FDLO							
	W								
0x000C RESERVED3 <sup>1</sup>	R	0	0	0	0	0	0	0	0
	W								
0x000D RESERVED4 <sup>1</sup>	R	0	0	0	0	0	0	0	0
	W								
0x000E RESERVED5 <sup>1</sup>	R	0	0	0	0	0	0	0	0
	W								
0x000F RESERVED6 <sup>1</sup>	R	0	0	0	0	0	0	0	0
	W								

= Unimplemented or Reserved

**Figure 17-3. Flash Register Summary**

1. Intended for factory test purposes only.



## 17.4.1.4 Illegal Flash Operations

### 17.4.1.4.1 Access Error

The ACCERR flag in the FSTAT register will be set during the command write sequence if any of the following illegal Flash operations are performed causing the command write sequence to immediately abort:

1. Writing to the Flash address space before initializing the FCLKDIV register
2. Writing a misaligned word or a byte to the valid Flash address space
3. Writing to the Flash address space while CBEIF is not set
4. Writing a second word to the Flash address space before executing a program or erase command on the previously written word
5. Writing to any Flash register other than FCMD after writing a word to the Flash address space
6. Writing a second command to the FCMD register before executing the previously written command
7. Writing an invalid command to the FCMD register
8. Writing to any Flash register other than FSTAT (to clear CBEIF) after writing to the FCMD register
9. The part enters stop mode and a program or erase command is in progress. The command is aborted and any pending command is killed
10. When security is enabled, a command other than mass erase originating from a non-secure memory or from the background debug mode is written to the FCMD register
11. A 0 is written to the CBEIF bit in the FSTAT register to abort a command write sequence.

The ACCERR flag will not be set if any Flash register is read during the command write sequence. If the Flash array is read during execution of an algorithm (CCIF=0), the Flash module will return invalid data and the ACCERR flag will not be set. If an ACCERR flag is set in the FSTAT register, the Flash command controller is locked. It is not possible to launch another command until the ACCERR flag is cleared.

### 17.4.1.4.2 Protection Violation

The PVIOL flag in the FSTAT register will be set during the command write sequence after the word write to the Flash address space if any of the following illegal Flash operations are performed, causing the command write sequence to immediately abort:

1. Writing a Flash address to program in a protected area of the Flash array (see [Section 17.3.2.5](#)).
2. Writing a Flash address to erase in a protected area of the Flash array.
3. Writing the mass erase command to the FCMD register while any protection is enabled.

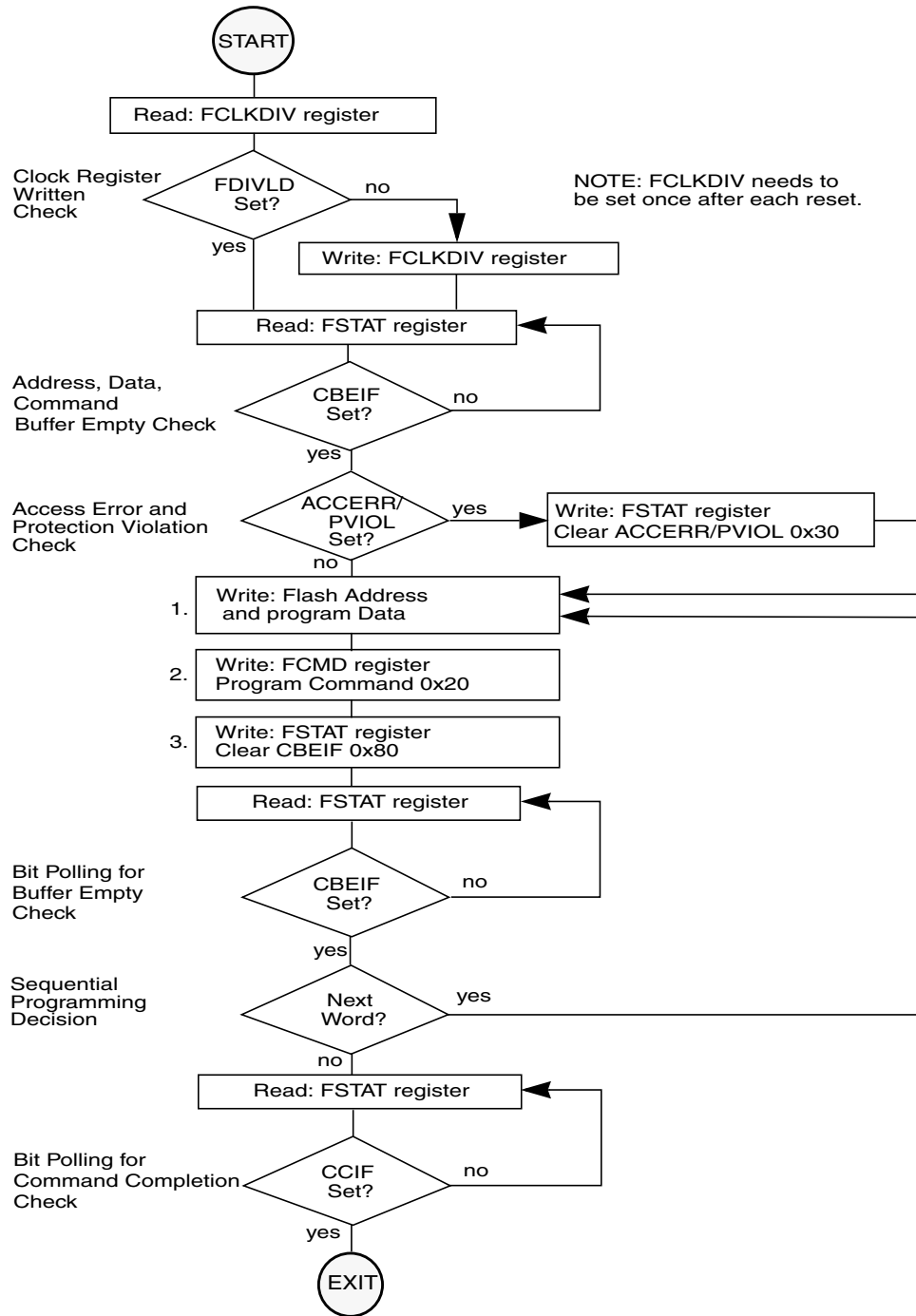
If the PVIOL flag is set, the Flash command controller is locked. It is not possible to launch another command until the PVIOL flag is cleared.

Table 19-3. Flash Array Memory Map Summary

MCU Address Range	PPAGE	Protectable Low Range	Protectable High Range	Array Relative Address <sup>(1)</sup>
0x0000–0x3FFF <sup>(2)</sup>	Unpaged (0x3D)	N.A.	N.A.	0x14000–0x17FFF
0x4000–0x7FFF	Unpaged (0x3E)	0x4000–0x43FF 0x4000–0x47FF 0x4000–0x4FFF 0x4000–0x5FFF	N.A.	0x18000–0x1BFFF
0x8000–0xBFFF	0x3C	N.A.	N.A.	0x10000–0x13FFF
	0x3D	N.A.	N.A.	0x14000–0x17FFF
	0x3E	0x8000–0x83FF 0x8000–0x87FF 0x8000–0x8FFF 0x8000–0x9FFF	N.A.	0x18000–0x1BFFF
	0x3F	N.A.	0xB800–0xBFFF 0xB000–0xBFFF 0xA000–0xBFFF 0x8000–0xBFFF	0x1C000–0x1FFFF
0xC000–0xFFFF	Unpaged (0x3F)	N.A.	0xF800–0xFFFF 0xF000–0xFFFF 0xE000–0xFFFF 0xC000–0xFFFF	0x1C000–0x1FFFF

1. Inside Flash block.

2. If allowed by MCU.



**Figure 19-26. Example Program Command Flow**

### 20.4.1.2 Command Write Sequence

The Flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the ACCERR and PVIOL flags in the FSTAT register must be clear and the CBEIF flag should be tested to determine the state of the address, data, and command buffers. If the CBEIF flag is set, indicating the buffers are empty, a new command write sequence can be started. If the CBEIF flag is clear, indicating the buffers are not available, a new command write sequence will overwrite the contents of the address, data, and command buffers.

A command write sequence consists of three steps which must be strictly adhered to with writes to the Flash module not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the Flash array memory.
2. Write a valid command to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the command.

The address written in step 1 will be stored in the FADDR registers and the data will be stored in the FDATA registers. When the CBEIF flag is cleared in step 3, the CCIF flag is cleared by the Flash command controller indicating that the command was successfully launched. For all command write sequences, the CBEIF flag will set after the CCIF flag is cleared indicating that the address, data, and command buffers are ready for a new command write sequence to begin. A buffered command will wait for the active operation to be completed before being launched. Once a command is launched, the completion of the command operation is indicated by the setting of the CCIF flag in the FSTAT register. The CCIF flag will set upon completion of all active and buffered commands.

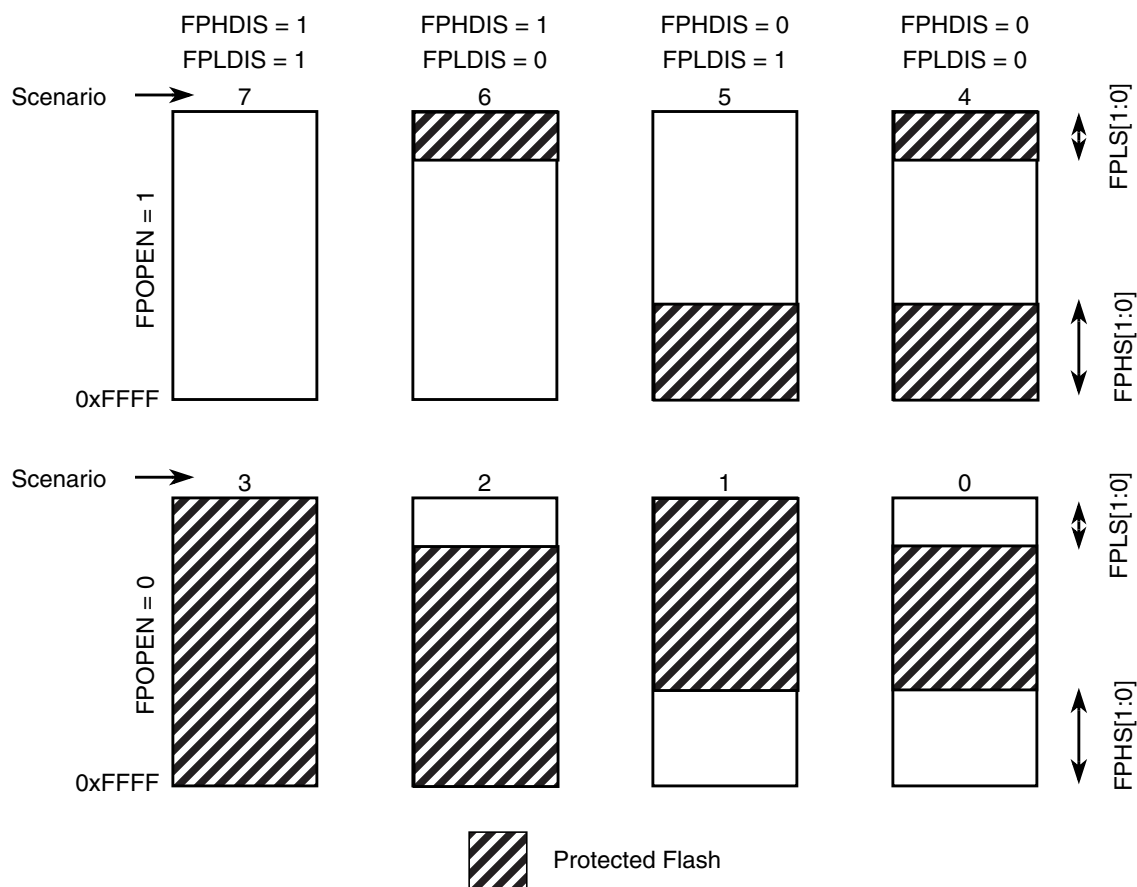


Figure 21-9. Flash Protection Scenarios

### 21.3.2.5.1 Flash Protection Restrictions

The general guideline is that protection can only be added, not removed. All valid transitions between Flash protection scenarios are specified in [Table 21-12](#). Any attempt to write an invalid scenario to the FPROT register will be ignored and the FPROT register will remain unchanged. The contents of the FPROT register reflect the active protection scenario.

Table 21-12. Flash Protection Scenario Transitions

From Protection Scenario	To Protection Scenario <sup>(1)</sup>							
	0	1	2	3	4	5	6	7
0	X	X	X	X				
1		X		X				
2			X	X				
3				X				
4				X	X			
5			X	X	X	X		